Adaptive Management of Computing and Network Resources for Spacecraft Systems [1]

Ryan Detter[32], Lonnie R. Welch[3], Barbara Pfarr[4], Brett Tjaden,[3] and Eui-Nam Huh[3]

Section 1 Introduction

It is likely that NASA's future spacecraft systems will consist of distributed processes which will handle dynamically varying workloads in response to perceived scientific events, spacecraft environments, spacecraft anomalies and user commands. Since these various states can not be determined prior to deployment, a method to dynamically adapt the system to handle the dynamic conditions was needed. To address this, we are evolving the *DeSiDeRaTa* adaptive resource management approach to enable reconfigurable ground and space information systems. This approach embodies a set of middleware for adapting resource allocations, and a framework for reasoning about real-time performance of distributed application systems. We are integrating the DeSiDeRaTa adaptive resource management system with a ground-based testbed called ITOS, Integrated Test and Operations System, that will enable NASA to perform early evaluation of adaptive resource management techniques without the expense of deploying the systems in space.

NASA is evolving the Principal Investigator's DARPA-funded *DeSiDeRaTa* adaptive resource management approach to enable reconfigurable real-time ground and space information systems. The benefits of the effort are numerous, including the ability to use sensors in new ways not anticipated at design time; the production of information technology that ties intra-constellation networks together; the accommodation of greater numbers of missions with fewer resources. For a comparison of DeSiDeRaTa to related work see Section 2.1. As detailed in Section 2.2, DeSiDeRaTa embodies a set of middleware mechanisms for adapting resource allocations, and a framework for reasoning about the real-time performance of distributed application systems. Section 2.2 also describes how the framework and middleware are being extended to accommodate the dynamic aspects of intra-constellation network topologies, and the complete real-time path from the instrument to the user. Our plans for assessment and prototyping of our approach for ITOS are described in Section 2.3. The first step of the plan, modeling of the ITOS system, is detailed in Section 2.4, describing the process used as well as an explanation of the ITOS path model (Figure 3).

[1] Funded by NASA Office of Earth Science, NASA Research Announcement 99-OES-08.
[2] This work was performed in part while Ryan Detter was in a co-op position in the Real-Time Software Engineering Branch at the NASA Goddard Space Flight Center.
[3] Laboratory for Intelligent, Real-Time, Secure Systems; School of Electrical Engineering and Computer Science; Ohio University; {welch|tjaden|detter|huh}@ohio.edu.
[4] Real-Time Software Engineering Branch; NASA Goddard Space Flight Center; barbara.pfarr@gsfc.nasa.gov.

2.1 Background and previous work in this problem area

The majority of real-time computing research has focused on the scheduling and analysis of real-time systems whose timing properties and execution behavior are known *a priori* (i.e., their derivation is based, in part, on theory instead of on experience or on experiment). However, there are numerous applications that must execute in highly dynamic environments (such as Earth and Space Science Missions), thereby precluding *accurate* characterization of the applications' properties by static models. In such contexts, temporal and execution characteristics can only be determined accurately by empirical observation or experience (i.e., *a posteriori*).

In most real-time computing models, the execution time of a "job" is used to characterize workload *a priori* as an integer "worst-case" execution time (WCET) [9, 16, 11, 3]. While [11] establishes the utility of WCET-based approaches by listing their domains of successful application, others [8, 6, 5, 7, 14, 15, 13, 12, 10, 1, 2, 4] cite the drawbacks, and in some cases the inapplicability, of the approaches in certain domains. In [15, 8, 5, 1] it is mentioned that characterizing workloads of real-time systems using *a priori* worst-case execution times can lead to poor resource utilization, particularly when the difference between WCET and normal execution time is large. Furthermore, it is stated in [12, 1] that accurately measuring WCET is often difficult and sometimes impossible.

DeSiDeRaTa project [17] is addressing these shortcomings. One of the fundamental innovations of the DeSiDeRaTa project is the dynamic path paradigm, which is employed for modeling and resource management of large-grain, distributed real-time mission-critical systems. The approach was experimentally validated on a benchmark system, and within an experimental Navy distributed computing system. Experimental results show the effectiveness of the approach for specification of real-time Quality of Service (QoS), detection and diagnosis of QoS failures, and restoration of acceptable QoS via reallocation of distributed computer and network resources.

2.2 The Adaptive Resource Management Approach

To support The Earth Science Vision and ITOS, NASA is evolving the DeSiDeRaTa adaptive resource management approach to *control* all essential processes and to *coordinate* the data flow through ITOS. This section describes the DeSiDeRaTa approach for managing distributed computing and network resources in order to meet the real-time performance goals of mission-critical systems. It also describes how we are extending the approach to accommodate the dynamic aspects of intra-constellation network topologies, and the complete real-time path from the instrument to the user.

The DeSiDeRaTa project is producing technology to enable the engineering of the emerging generation of distributed real-time systems. Such systems have rigorous QoS objectives. They must behave in a dependable manner, respond to important events in a timely fashion and provide continuous availability within harsh environments. Furthermore, resources should be utilized in an efficient manner, and scalability must be provided to address the ever-increasing complexity of scenarios that confront such systems. To provide the desired QoS in such a context, efforts are focusing on the following aspects: i) QoS specification, ii) QoS metrics, iii) dynamic QoS management,

and iv) benchmarking. The project is called DeSiDeRaTa, for its applicability to Dynamic, Scalable, Dependable, Real-Time systems.

The DeSiDeRaTa project is providing innovative QoS management technology that incorporates knowledge of needs in the distributed real-time mission-critical systems domain. The technology includes a specification language and dynamic QoS management software that support dynamic, path-based systems. Additionally, domain knowledge is being used to produce a dynamic real-time mission-critical benchmark suite.

DeSiDeRaTa technology supports the dynamic path concept for automated QoS assessment and resource allocation. A dynamic path is a very large-gain entity, which typically consists of sensors, actuators, and control software for filtering, evaluating, and acting. The paths may have timing constraints, may have widely varying dynamic behavior, and may be scalable and fault tolerant. A generic path design pattern contains: (1) a data source and/or an event source, (2) a data (and/or event) stream and (3) a data (and/or event) consumer. The data/event *source* produces a *stream* of data/events, which cause the consumer to perform processing. A data/event source is typically one or more sensors, but may also be a clock or a software entity. A *consumer* often contains one or more actuators. Each datum or event is evaluated by a data consumer, to decide whether the actuators should perform actions.

Path-level QoS specification is used by the adaptive resource allocator to determine if the current configuration is achieving the desired QoS and to assist in selecting new configurations to improve QoS. This is significantly different than other approaches to assessment in the sense that it is performed dynamically, and is performed at a much larger granularity. Moreover, DeSiDeRaTa differs from previous work in that it accounts for the complex features of dynamic real-time systems. These features include previously overlooked issues with respect to granularity, variable periods, sporadic processes, priorities, fault management and scalability.

The logical architecture of the DeSiDeRaTa QoS management software is shown in Figure 1. It behaves as follows. The application programs of real-time control paths send time-stamped events to the *QoS metrics* component, which calculates path-level QoS metrics and sends them to the *QoS monitor*. The monitor checks for conformance of observed QoS to required QoS, and notifies the *QoS diagnosis* component when a QoS violation occurs. The diagnoser notifies the *action selection* component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS. Action selection ranks the recommended actions, identifies redundant actions, and forwards the results to the *allocation analysis* component; this component consults *resource discovery* for host and LAN load index metrics, and determines an efficient method for allocating the hardware resources to perform the actions, and requests that the actions be performed by the *allocation enactment* component.

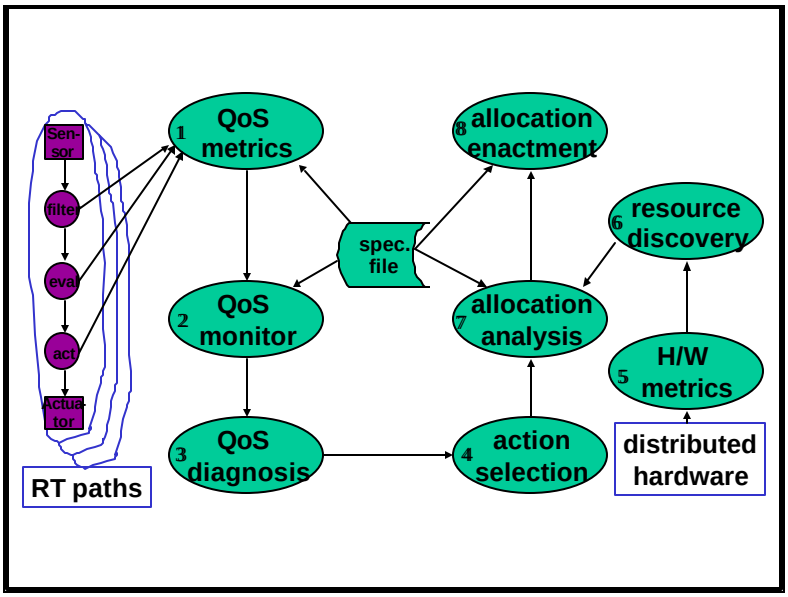

Figure 1. Logical architecture of the resource and QoS management software.

The dynamic path paradigm is ideal for modeling the complete real-time path from the instrument to the user. Current DeSiDeRaTa resource management strategies assume that each path can have a variable workload – the number of data stream elements and the event arrival rate can vary. Both of which reflect features of the constellations of the future. However, other types of variability cannot be modeled at present. Thus, NASA is generalizing the adaptive resource management approach to accommodate dynamic numbers of data streams and dynamic types of data stream elements.

DeSiDeRaTa's model of computing and network resources also provides a solid basis for management of space-based information systems. It can represent static resource features (such as the speed of a CPU), and static network topologies. Additionally, it has sophisticated metrics for dynamic resource load indexes that are applicable in heterogeneous resource environments.

Currently, we are collaborating with the Resource Monitoring System (Remos) team at Carnegie Mellon University to adapt their network monitoring software to our resource management needs. Remos provides a query-based interface that will allow DeSiDeRaTa to assess the topology of the network resources and their available capacity. Once network monitoring using Remos is integrated, the resource manager will be extended to incorporate network metrics into the QoS monitoring, diagnosis, action selection, and allocation enactment component - allowing DeSiDeRaTa complete control of all aspects of a real-time path.

2.3 Prototyping a reconfigurable computing strategy for NASA's Integrated Test and Operations System (ITOS)

We are demonstrating the feasibility of adaptive resource management for a ground-based system currently used by several NASA satellite systems. We are showing the applicability of our approach for dynamically managing the information systems of satellites. Moreover, we are providing a ground-based testbed that is enabling NASA to

perform early evaluation of constellation management techniques without the expense of first deploying them in space.

The effectiveness of adaptive resource management is being demonstrated in a ground based real-time system called ITOS. ITOS, the Integrated Test and Operations System, is a suite of software developed by a group in the Real-time Software Engineering Branch at the Goddard Space Flight Center for control of spacecraft and spacecraft components during development, test, and on-orbit operations. Figure 2 depicts a typical ITOS data system configuration for spacecraft integration and test (I&T) or on-orbit operations. In the future, some portion of this configuration will likely be contained within an autonomous constellation of satellites.

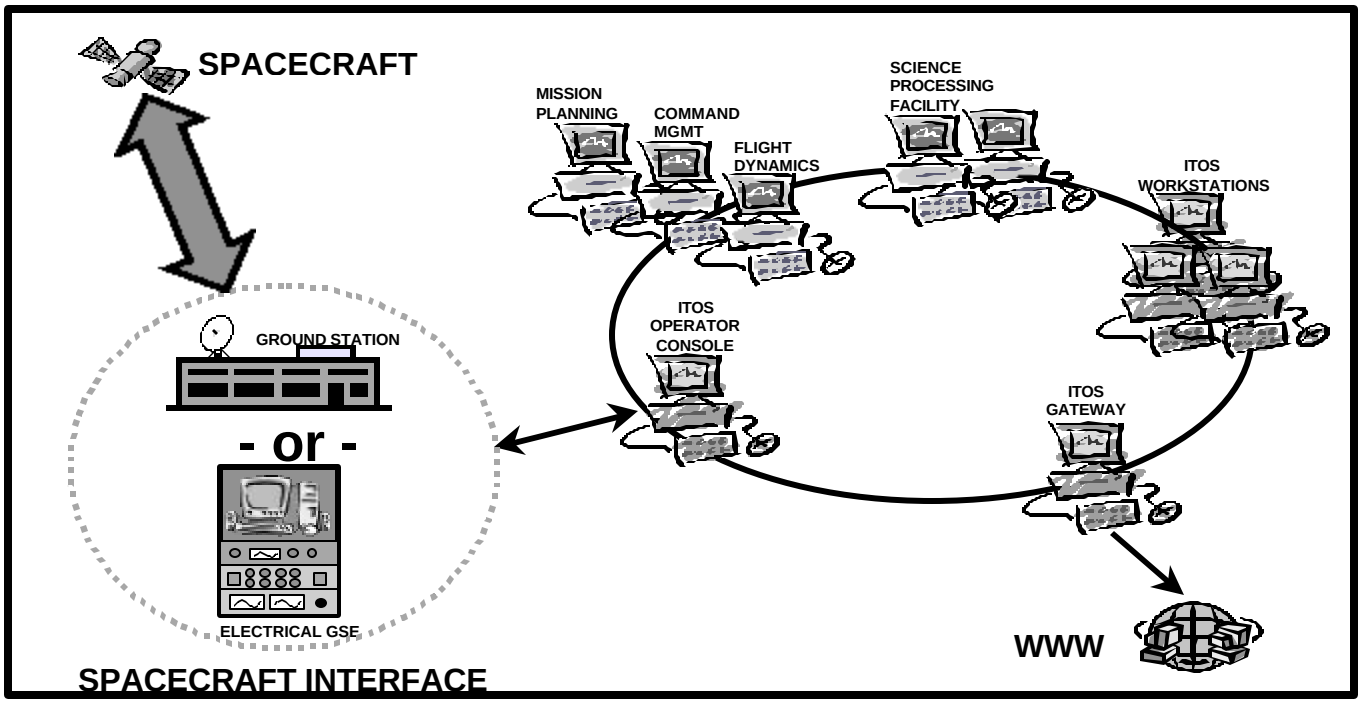

Figure 2. ITOS for Spacecraft Integration/ On-Orbit Operations

The ITOS configuration consists of a cluster of workstations interconnected over a local area network. Each workstation runs the complete ITOS software, with one designated as the primary operator console. This console receives telemetry data and sends commands to and from the spacecraft interface over an IP Ethernet connection. The primary console feeds the telemetry data it receives from the spacecraft interface to all other ITOS workstations. Each ITOS workstation unpacks the telemetry packets and performs data processing tasks such as limit checking, engineering unit conversions, and configuration monitoring. ITOS has an event subsystem, which recognizes spacecraft events, logs them, and forwards them to operators or external programs for processing. In addition, the primary ITOS console can distribute telemetry data via an IP Ethernet to other systems attached to the local area network; for example, the Science Processing Facility, Command Management System, Flight Dynamics System, and Mission Planning System. This approach is analogous with the one to be taken within space-based sensor webs.

Typically, ITOS installations in mission operations or I&T are set up on private

networks. One ITOS computer will, however, be connected to the Internet as a server for ITOS's Java-language, web-based telemetry and event displays.

ITOS was a good candidate for adaptive resource management. First, the workload of an ITOS-based system is environment-dependent, and thus varies at run-time in several ways, including:

- The telemetry data rate varies
- The number and types of displays vary
- The number of telemetry streams varies
- The types of elements within a particular telemetry stream varies
- The number of workstations being served with telemetry data varies

Another reason that ITOS was a good candidate is that it has the ability to be dynamically adapted to handle varying workloads via utilization of a pool of distributed computing resources. However, the most compelling reason NASA decided to apply adaptive resource management to ITOS was because it provides significant risk mitigation for the Earth Science Vision.

To show proof-of-concept, the resource management approach described in Section 2.2 is being applied to ITOS. This application of the approach to ITOS is being performed in the following steps:

1. Model the sensors, actuators, applications, and communication relationships within ITOS
2. Construct a model of the dynamic real-time paths of ITOS (Section 2.4)
3. Identify fault tolerance requirements
4. Insert probes into ITOS applications (e.g., for events & resource usage)
5. Specify the above properties in D-Spec (the DeSiDeRaTa QoS specification language)
6. Obtain resource usage profiles for the ITOS applications and paths
7. Make applications scaleable & fault tolerant
8. Deploy adaptive resource management for the application system

The effectiveness of the resource management approach for ITOS will be determined via extensive experimentation. The components of ITOS will be distributed over a collection of computers, and Network Time Protocol (NTP) will be used to provide global time. Dynamic scenarios will be run and real-time QoS will be recorded. The scenarios will be challenging enough to force resource reallocations. Collecting several assessment metrics that we have developed will assess the quality of the adaptive resource management strategy. Ideally, QoS will be improved with each reallocation action. Thus, we will measure the *absolute* and *percentage improvement* in QoS per reallocation action. Other metrics that we will gather are (1) *QoS violation rate* (QVR) – the number of QoS violations per unit of time, (2) the *sensitivity* of chosen allocations to increased load, and the time to perform resource reallocations.

2.4 ITOS Model and Path Behavior

Our first step in modeling ITOS was to define the actuators, sensors, applications, and communication relationships within ITOS. We determined the data and loads vary

depending on the spacecraft I/O data rate, types and number of displays, types of commands, and the number of event messages generated. Since these characteristics only effected the telemetry, command, and event systems we decided to make these our critical paths. It was determined that each path had a sensor that received data, sending it to the components, or applications, in the path for processing, then sending data to the path actuator for action. We defined the sensors and actuators in each path as follows:

	Sensor	**Actuator**
Telemetry	Spacecraft	STOL, Displays, fop
Command	STOL	Spacecraft
Event	ITOS components	dsp_evtdsp , evt_forward, dsp_evtlog

The communication relationships between the applications, sensors, and actuators can be seen in Figure 3.

After studying the sensors, actuators, applications, and communication relationships within ITOS, we constructed a model of the dynamic real-time paths of ITOS. In evaluating the components and subsystems within ITOS we were able to break ITOS into four systems and two subsystems.

ITOS includes the event, telemetry, command, and STOL systems. The event system is used for the logging of every action done in the ITOS system. This log is generated in both a display window, for the operators reference, as well as in a file format that can be accessed for future reference. The telemetry system is utilized whenever data, of various types, is received from the space system, or any other system under ITOS control. It takes care of receiving the frames from the spacecraft, unpacking the frames into packets, unpacking the packets into useful data, and installing the data into the database where it can be accessed by mnemonic variable names by various ITOS applications. The command system controls the sending of various data and commands to the spacecraft as well as verification of command receipt. It creates the frames from packets, sequences the frames, sends the command to the external destination, and performs closed-loop command verification, in accordance with the CCSDS COP-1 protocol, to ensure the spacecraft has received the commands. The STOL, Spacecraft Test and Operations Language, system is the backbone of the ITOS system, used to control the configuration and execution of all ITOS systems. For non-trivial spacecraft operations, multiple STOL directives are combined to create a procedure that can be executed by the STOL system.

We have delineated two subsystems in ITOS, display and frame_sorter, which make up part of the telemetry system. The display subsystem consists of sequential prints, plots, configuration monitors, and a server that enables access to ITOS telemetry values through a network. The frame_sorter subsystem extracts CCSDS packets from a stream of CCSDS frames. This subsystem also has the capability to handle TDM frames, CCSDS packets, as well as other data formats. The normal CCSDS frames to CCSDS packet operation is first done by sorting the frames into virtual channels (VC's). The frame_sorter then manipulates the VC's, separating them into their respective data packets. The packets are then output to enable use by the telemetry system as needed. The frame_sorter is also the device used to archive all of the data. This archiving can be done both in frame and packet formats, to be used for later reference.

When integrated together and connected with data and communication paths, we get the general ITOS system model, as shown in figure 3. These paths show the data and communication flow between each component of ITOS. The subsystems and systems include the following components (in the order of normal flow):

- Frame_sorter subsystem: frame_input, vc_sorter, pkt_extract, archive, output
- Display subsystem: ODB, dsp_mnepage, dsp_seqprt, dsp_plot, eqn_cfgmon, dp_server
- Telemetry system: frame_sorter subsystem, tlm_client, ODB, display subsystem
- Command system: stol, cmd_load, ODB, fop_mux, fop, cmd_transmit
- STOL system: stol_fifo, stol_wkp, stol, stol_i/f, ODB
- Event system: event_fifo, dsp_event, dsp_evtlog, dsp_evtdsp, evt_forward

To enable control of these systems, the behavior of the three paths, telemetry, command, and event, needed to be determined. The flow of data, path activation, and size of data behaviors needed to be specified. Data in ITOS is only generated after an event occurs; once the event occurs the flow is continuous. Therefore the behavior of the data flow was neither continuous nor transient, but quasi-continuous. The event that activates the data flow was found to be a STOL command while the data is dynamic, since each mnemonic, event message, and command varies in size.

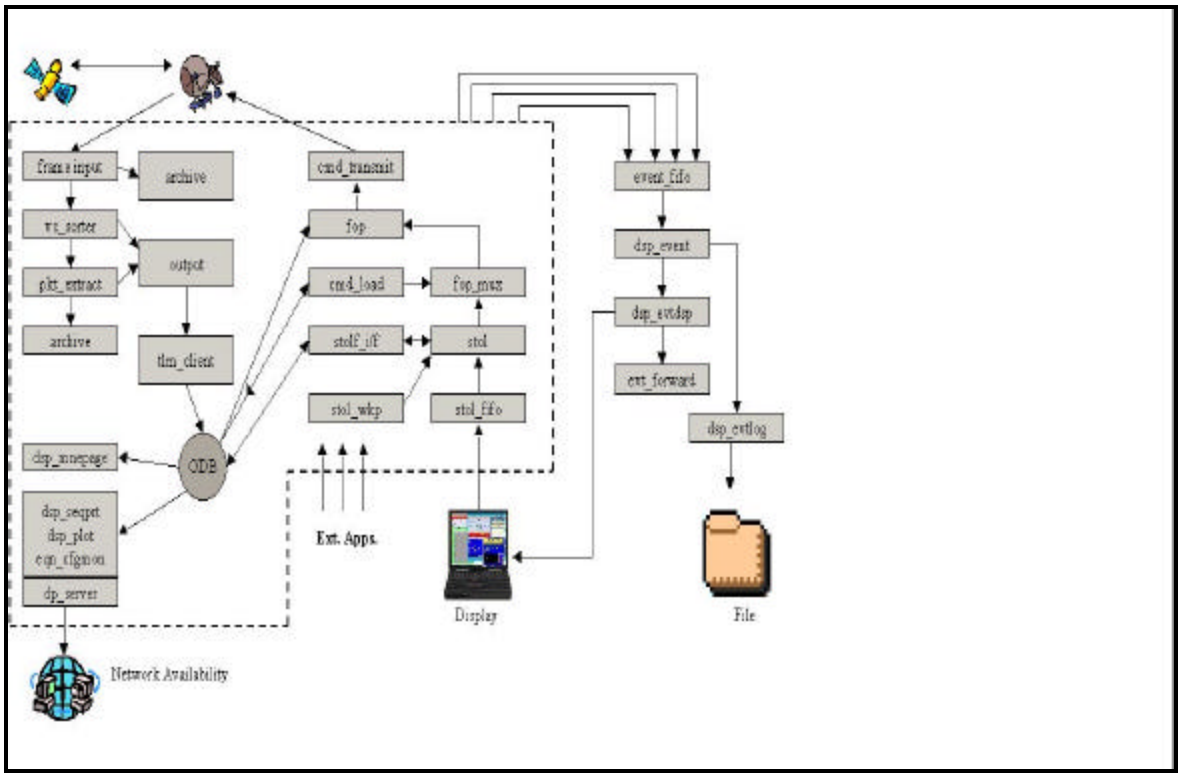

Figure 3. Path model of ITOS

Section 3 Conclusions and Future Work

This paper has presented the concept of DeSiDeRaTa's adaptive resource management approach and it's applications in spacecraft applications. Since NASA's future spacecraft will work in dynamic environments and with dynamic workloads, we are evolving the resource management approach to accommodate these characteristics in an initial test-bed: ITOS. The dynamic and distributed attributes of ITOS are being met by DeSiDeRaTa's use of a set of middleware for adapting resource allocations, and a framework for reasoning about the real-time performance of distributed application systems.

While the majority of real-time computing research has focused on the scheduling and analysis of real-time systems who's timing properties and execution behavior are known *a priori* (section 2.1), the DeSiDeRaTa resource management approach focuses on applications that must execute in highly dynamic environments. One of the fundamental ways we achieve this is by the innovation of the dynamic path paradigm (section 2.2). The project is producing technology to enable the engineering systems containing rigorous QoS objectives such as dependability, timely event response, unlimited continuous availability, efficient resource utilization, and scalability.

The use of ITOS as the initial test-bed for dynamic spacecraft systems and the adaptive resource management approach was inherent since its workload is environment-dependent and varies at run-time in several ways (section 2.3). Another reason that ITOS was a good candidate is for its ability to dynamically adapt to handle varying workloads with the use of a distributed computing resource pool.

The first step of integrating the resource manager with ITOS was described (section 2.4) and we are continuing with the remaining steps. The ITOS fault tolerance requirements will be determined, probes will be inserted, spec files will be written, usage profiles will be defined, and applications will be made scaleable as well as fault tolerant. Once completed, the DeSiDeRaTa controlled ITOS will go through extensive experimentation to determine the adaptive resource management approach's effectiveness, therefore allowing NASA to perform early evaluation of dynamic systems without the expense of deploying spacecraft.

Section 4 References

[1] L. Abeni and G. Buttazzo, “Integrating multimedia applications in hard real-time systems,” in *Proceedings of the 19th IEEE Real-Time Systems Symposium,* 3-13, IEEE Computer Society Press, 1998.

[2] A. Atlas and A. Bestavros, “Statistical rate monotonic scheduling,” in *Proceedings of the 19th IEEE Real-Time Systems Symposium,* 123-132, IEEE Computer Society Press, 1998.

[3] T.P. Baker, “Stack-based scheduling of realtime processes,” *Journal of Real-time Systems,* **3(1)**, March 1991, 67-99.

[4] S. Brandt, G. Nutt, T. Berk and J. Mankovich, “A dynamic quality of service middleware agent for mediating application resource usage,” in *Proceedings of the 19th IEEE Real-Time Sys. Symposium,* 307-317, IEEE Computer Society Press, 1998.

[5] D. Haban and K.G. Shin, "Applications of real-time monitoring for scheduling tasks with random execution times," *IEEE Transactions on Software Engineering*, **16(12)**, December 1990, 1374-1389.
[6] F. Jahanian, "Run-time monitoring of real-time systems," in *Advances in Real-time Systems*, Prentice-Hall, 1995, 435-460, edited by S.H. Son.
[7] T.E. Kuo and A. K. Mok, "Incremental reconfiguration and load adjustment in adaptive real-time systems," *IEEE Transactions on Computers*, **46(12)**, December 1997, 1313-1324.
[8] J. Lehoczky, "Real-time queueing theory," in *Proceedings of the 17th IEEE Real-Time Systems Symposium*, 186-195, IEEE Computer Society Press, 1996.
[9] C.L. Liu and J.W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, **20**, 1973, 46-61.
[10] J.W.S. Liu, K.J. Lin, W.K. Shih, A.C. Yu, J.Y. Chung and W. Zhao, "Algorithms for scheduling imprecise computations," *IEEE Computer*, **24(5)**, May 1991, 129-139.
[11] L. Sha, M. H. Klein, and J.B. Goodenough, "Rate monotonic analysis for real-time systems," in *Scheduling and Resource Management*, Kluwer, 1991, 129-156, edited by A. M. van Tilborg and G. M. Koob.
[12] D.B. Stewart and P.K. Khosla, "Mechanisms for detecting and handling timing errors," *Communications of the ACM* , **40(1)**, January 1997,87-93.
[13] H. Streich and M. Gergeleit, "On the design of a dynamic distributed real-time environment," in *Proceedings of the 5th International Workshop on Parallel and Distributed Real-Time Systems*, 251-256, IEEE Computer Society Press, 1997.
[14] J. Sun and J.W.S. Liu, "Bounding completion times of jobs with arbitrary release times and variable execution times," in *Proceedings of the 17th IEEE Real-Time Systems Symposium*, 2-11, IEEE Computer Society Press 1996.
[15] T.S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.C. Wu and J.W.S. Liu, "Probabilistic performance guarantee for real-time tasks with varying computation times," in *Proceedings of the 1st IEEE Real-Time Technology and Applications Symposium*, 164-173, IEEE Computer Society Press, 1995.
[16] L. R. Welch, A. D. Stoyenko, and T. J. Marlowe, "Modeling resource contention among distributed periodic processes specified in CaRT-Spec," *Control Engineering Practice*, **3(5)**, May 1995, 651-664.
[17] L. R. Welch, B. Ravindran, B. Shirazi and C. Bruggeman, "Specification and analysis of dynamic, distributed real-time systems," in *Proceedings of the 19th IEEE Real-Time Systems Symposium*, 72-81, IEEE Computer Society Press, 1998.
[18] G. E. Prescott, S. A. Smith and K. Moe, "Real-Time Information System Technology Challenges for NASA's Earth Science Enterprise," in *Proceedings of the International Workshop on Real-Time Mission-Critical Systems*, Dec. 1999.